IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

ATTY. DOCKET NO.: RAL9-00-0029

In re Application of: §

CLAUDE BASSO, ET AL. § Examiner: **Diem K. Cao**

Serial No.: **09/546,133** §

Filed: ***April 10, 2000*** § Art Unit: **2126**

For: **NETWORK PROCESSOR SERVICES ARCHITECTURE THAT IS PLATFORM AND OPERATING SYSTEM INDEPENDENT** §

APPEAL BRIEF UNDER 37 C.F.R. 1.192

Mail Stop Briefs - Patents
Commissioner for Patents
Washington, D.C. 20231

Sir:

This Appeal Brief is submitted in triplicate in support of an Appeal of the Examiner's final rejection of Claims 1-20 in the above-identified application. A Notice of Appeal was filed in this case on July 6, 2004. Please charge the fee of $330.00 due under 37 C.F.R. § 1.17(c) for filing the brief, as well as any additional required fees, to IBM Deposit Account No. 50-0563.

Certificate of Transmission/Mailing

I hereby certify that this correspondence is being facsimile transmitted to the USPTO at 703-872-9306 or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450 on the date shown below.

Typed or Printed Name: PIA LORENZANA *Date:* September 7, 2004 *Signature:*

REAL PARTY IN INTEREST

The real party in interest in the present Appeal is International Business Machines Corporation, the Assignee of the present application as evidenced by the Assignment recorded at reel 010967 and frame 0863 *et. seq*.

RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellants, the Appellants' legal representative, or assignee, which directly affect or would be directly affected by or have a bearing on the Board's decision in the pending appeal.

STATUS OF CLAIMS

Claims 1-5, 7-11, 13-16, 18 and 20 stand finally rejected by the Examiner as noted in the Advisory Action dated June 9, 2004.

STATUS OF AMENDMENTS

Appellants' Amendment B, filed on May 5, 2004, was entered by the Examiner.

SUMMARY OF THE INVENTION

As described by the summary (*see* page 5) and other sections of the specification, Appellants' invention provides a network processor services architecture (hereinafter "NPSA") that is platform and operating system (OS) independent (*see* page 9, lines 11-29; page 18, lines 11 – page 19, line 4 describing OS independence and portability). That is, within a NPSA, a number of portable, exchangeable, and/or scalable functional components are provided that are adaptable to various types of processor architecture and OS and enable customization and expansion of available network services (*see* page 14, line 19 - page 15, line 19, which also describes the functionality of external APIs). The functional components may be combined to provide a control processor's device driver and include lower-level APIs and external APIs that carry out specific network processor functions (*see* page 19, lines 6-19) required by a control processor to communicate with and direct the operations of the network processor (*see* page 10, lines 7-15, 21-26).

ISSUES

The primary issue for appeal is whether Examiner's rejection of Appellants' claims under 35 U.S.C. § 103(a), as being unpatentable over *Flory*, et al. (U.S. Patent No. 6,009,476) in view of *Chang*, et al. (U.S. Patent No. 6,604,136 B1) is well founded.

GROUPING OF THE CLAIMS

For purposes of this Appeal, all claims stand or fall together as a single group.

ARGUMENT

Examiner's rejection of Appellants' exemplary claim 1 and other dependent claim features as being unpatentable over *Flory* in view of *Chang* is not well founded and should be reversed.

Appellants hereby incorporate by reference the arguments proffered in previously filed Amendment A and Amendment B. With respect to the relevant arguments provided in these two Amendments, Appellants specifically reiterate that the combination of *Flory* and *Chang* fails to suggest to one skilled in the art the following features recited by Appellants' exemplary claims:

(1) "..., loading one or more of a plurality of functional components ... provide the desired **network processor functionality** and wherein each of said plurality of functional components are independently selectable and provide a respective one of a plurality of **network processor services**;"

(2) "providing at least one utility ... provides an **OS-independent communication interface** for said plurality of functional components and enables **bi-directional communication**; wherein said providing step further provides **a translation utility**, which **translates all incoming and outgoing service requests into a common network processor language** to permit seamless connection and correspondence between said one or more network processors, said operating system, and each of said functional components to enable handling of network packets;" and

(3) "in response to a receipt of a packet at said control system, ... said packet is **decoded into a common code understandable by said OS** and said ... functional components, ...

wherein said utility also converts all low and high level APIs of various components into a specific communication type utilized by the network processor functionality" (emphases added).

Further, as recited by dependent claims, Appellants' invention further provides that:

(1a) the desired network processor functionality includes "external APIs, low level APIs, and physical transport interfaces of a device driver;" (Cl. 2); and

(1b) "loading a customer definable service component within said external (APIs) that includes a customer desired network service, which is operable within said network processor services architecture" (Cl. 3; emphases added)

These claim features indicated that Appellants' claims are specifically directed to communication among the various processor components and functional components that connect to and interact with the network processor services architecture, some of which are external components with external APIs. The claims also provide for a translation of service requests into a common network processor language to permit correspondence between network processors.

A. Implementation within network processor services architecture having multiple network processors

Appellants' claimed invention provides a "network processor services architecture," which is provided within a "distributed processing system" having multiple processors interconnected via some external interconnection mechanism (e.g., a switch). One skilled in the art would recognize that a network processor services architecture within a distributed system necessarily contains multiple network processors that comprise specific operating protocols and interconnection methods, as well as unique process control functionality. While certain features of the invention are described as being implemented by "one or more" network processors, however, it is clear that a claim reference to communication between processors necessarily requires at least two network processors to communicate.

Flory, in contrast, provides a description of a single/individual computer system with a device driver that includes a device driver library with selectable execution contexts (*see* Figures

1 and 2 and descriptions thereof). As such, all functional features provided by *Flory* reference the single system environment, which is inherently different from the inter-operation involved with the distributed system environment (with multiple network processors) as described by Applicants' claimed invention. A skilled artisan would not mistake *Flory's* operations on a single computer with an internal processor and OS as being suggestive of the more complex configuration, functionality and operation on a distributed processing system that supports network processor services architecture.

Examiner correctly states at paragraph 7 of the Final Action that *Flory* does not teach a network processor and methods of handling packets within a distributed system environment as recited by Applicants' claims. Examiner references *Chang* for support of this feature.

There is no teaching within either reference to support combining the features of *Flory* with *Chang*. However, assuming support could be found for this combination, the combination still does not render Applicants' claimed invention unpatentable since the combination does not suggest several of the key features of Appellants' claims listed above.

B. External APIs with customer-definable service components

For example, Col. 7, line 45 - col.8, line 65 and col. 10, lines 21-30 of *Flory* are devoid of any reference of an external API or loading an external API or providing customer definable service components within external APIs. Those sections are very clear with respect to types of APIs supported, including an "OS API" and a "proprietary API". As recited by *Flory*, "[t]he shell module is the initial component of the device driver 50 loaded into the memory as part of the initialization of the operating system kernel 56 during system startup" (ll 59-62). *Flory* also states that the OS API "provides the operating system kernel 56 and any extensions 58 with entry points into the device driver 50" (ll 45-48). Given the very comprehensive descriptions of the types of APIs supported in both *Flory* and *Change*, it is clear that neither reference contemplates or suggests "loading ... external APIs... customer desired network services, which is operable within the network processor services architecture"

With respect to Claim 9, since *Flory* does not provide any network functionality (admitted by Examiner in paragraph 7) *Flory's* mention (at col.22, lines 27-46) of support for a new partial API, "either as newly defined by the operating system layer 54 or to support calls originated directly from an application 60" is clearly still within the single computer environment. Thus, nothing in that section of *Flory* suggests the external API utilized within a distributed system and the functionality associated therewith.

C. Translation utility and translation services

As correctly noted by Examiner, the features of Claims 12, which are now incorporated into the independent claim 7, are not taught (or suggested) by col. 22, lines 47- col. 23, line 6 of *Flory*. Those features are also not suggested by col. 7, line 48-63 of *Chang*. That section of *Chang* describes a host system connected to a network processor via a PCI bus. That section also discusses that other communication techniques may be used, and lists the various applications within which the network processor may be used. The section is, however, devoid of any reference or suggestion of translating service requests into a common network processor language ... one or more network processors...," as recited by Appellants' claims.

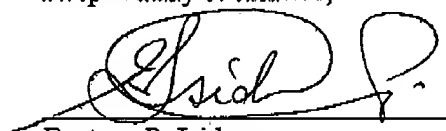
Clearly, neither reference teaches or suggests the functionality associated with the "translation utility" or the specific method of processing received packets including converting/translating the packets into a communication type utilized for communications amongst the network processors and other components. Other listed features are also not provided by the references or combination thereof.

For the above reasons, it is clear that the combination of *Flory* and *Chang* does not render unpatentable the features of Appellants' claimed invention. Examiner's rejection of Appellants' claim is therefore not well founded and should be reversed.

CONCLUSION

Appellants have pointed out with specificity the manifest error in the Examiner's rejections, and the claim language which renders the invention patentable over the combination of references. Appellants, therefore, respectfully requested that this case be remanded to the Examiner with instructions to issue a Notice of Allowance with respect to all pending claims.

Respectfully submitted,



Eustace P. Isidore
Registered with Limited Recognition (see attached)
Dillon & Yudell LLP
8911 North Capital of Texas Highway
Suite 2110
Austin, Texas 78759
512.343.6116

ATTORNEY FOR APPELLANTS

APPENDIX

1. In a distributed processing system, a method for providing scalable device driver services within a control system of a network processor services architecture, said method comprising the steps of:

in response to determining a desired network processor functionality, loading one or more of a plurality of functional components, wherein said one or more functional components provide the desired network processor functionality and wherein each of said plurality of functional components are independently selectable and provide a respective one of a plurality of network processor services;

providing at least one utility interposed between said one or more of said plurality of functional components and an operating system (OS) of said control system, wherein said at least one utility provides an OS-independent communication interface for said plurality of functional components and enables bi-directional communication between said network processor functionality and said OS;

wherein said providing step further provides a translation utility, which translates all incoming and outgoing service requests into a common network processor language to permit seamless connection and correspondence between said one or more network processors, said operating system, and each of said functional components to enable handling of network packets; and

in response to a receipt of a packet at said control system, first routing said packet through said utility, wherein said packet is decoded into a common code understandable by said OS and said one of said plurality of functional components, then handling said packet utilizing a selected one of said plurality of functional components, wherein said utility also converts all low and high level APIs of various components into a specific communication type utilized by the network processor functionality.

2. The method of Claim 1, wherein said loading step includes the step of loading external application programming interfaces (APIs), low level APIs, and physical transport interfaces of a device driver.

3. The method of Claim 2, further comprising the steps of:

loading a customer definable service component within said external (APIs) that includes a customer desired network service, which is operable within said network processor services architecture; and

dynamically expanding available services to include said customer definable service component on-the-fly.

4. The method of Claim 1, wherein said providing step includes the step of providing a bi-directional connection between said utility and said operating system, one or more network processors, and each of said functional components.

5. The method of claim 1, wherein said providing step further comprises the step of linking a system services utility to said operating system, wherein, said system services utility operates to allow each of said individual software coded components to communicate with said OS.

6. Canceled

7. A computer program product for providing scalable device driver services within a control system of a network processor services architecture comprising:

a computer readable medium; and

program instructions on said computer readable medium for:

implementing a plurality of individual software components that each provide a respective one of a plurality of network processor services;

providing at least one utility interposed between said plurality of individual software components and an operating system (OS) of said control system, that provide an OS independent communication interface for said plurality of individual software components;

wherein said program instructions for said providing step further includes program instructions for a translation utility, which translates all incoming and outgoing service requests into a common network processor language to permit seamless connection and correspondence between said one or more network processors, said

operating system, and each of said functional components to enable handling of network packets; and

in response to a receipt of a packet at said control system, first routing said packet through said utility, wherein said packet is translated into a code understandable by said OS and said one of said plurality of individual software components, then handling said packet utilizing one of said plurality of individual software components, wherein said utility also converts all low and high level APIs of various components into a specific communication type utilized by the network processor functionality.

8. The computer program product of Claim 7, wherein program instructions for said loading step includes program instructions for loading external application programming interfaces (APIs), low level APIs, and physical transport interfaces of a device driver.

9. The computer program product of Claim 8, further comprising program instructions for:

loading a customer definable service component within said external APIs that includes a customer desired network service, which is operable within said network processor services architecture; and

dynamically expanding available services to include said customer definable service component on-the-fly.

10. The computer program product of Claim 7, wherein said program instructions for said providing step includes instructions for providing a bi-directional connection between said utility and said operating system, one or more network processors, and each of said functional components.

11. The computer program product of claim 7, wherein said program instructions for said providing step further comprises instructions for linking a system services utility to said operating system, wherein, said system services utility operates to allow each of said individual software coded components to communicate with said OS.

12. Canceled

13. A control system of a network processor services architecture comprising:

a plurality of individually loadable functional components within a device driver of said control system that each represents a network processor service;

at least one utility for enabling each of said functional components to communicate with an operating system (OS) of said control system, wherein, in response to a receipt of a request by said OS to perform a network processor function, said utility translates said request into a call of a particular one of said plurality of functional components that administers said network processor services of one or more network processors, wherein said utility is located within lower level APIs and is communicatively coupled to external APIs, OS, software stack, and a network processor resource services, and wherein said utility also converts all low and high level APIs of various components into a specific communication type utilized by the network processor functionality;

wherein said utility includes a translation utility, which translates all incoming and outgoing service requests into a common network processor language to permit seamless connection and correspondence between said one or more network processors, said operating system, and each of said functional components to enable handling of network packets; and

processing hardware that executes code of said OS, said utility and said functional components.

14. The system of Claim 13, wherein said plurality of functional components includes external application programming interfaces (APIs), low level APIs, and physical transport interfaces of a device driver.

15. The system of Claim 14, further comprising a customer definable service component within said external (APIs) that includes a customer desired network service, which is operable within said network processor services architecture, wherein available services are dynamically expandable to include said customer definable service component on-the-fly.

16. The system of Claim 13, wherein said utility includes a system services utility coupled to said operating system that operates to allow each of said individual software coded components to communicate with said OS.

17. Canceled

18. A network processing services architecture comprising:

one or more network processors;

a switch fabric coupled to said one or more network processors; and

a control system, coupled to said one or more network processors, having a scalable device driver for controlling interactions with said one or more network processors, wherein said scalable device driver is comprised of a plurality of individual functional software components including a plurality of functional elements for network processing and a conversion utility to enable compatibility between said plurality of functional elements and said one or more network processors to effectively communicate and interact within any processor type and OS to effectuate the transfer of frames and control information between said control system and said one or more network processors.

19. Canceled

20. A device driver program for a distributed processing system comprising:

a plurality of system services programs providing high level bi-directional communication between a common code program and each of a network transport manager, a network resource manager and a control message formatter, said network transport manager providing low level bi-directional communication to a physical transport processor, said physical transport processor providing primitive bi-directional communication to one or more network processors.